ユークリッドアルゴリズム

情報数理学特別講義I 第10回

2025-11-06

前回の講義

- $t = 2, 3$ の2元BCH符号の誤り位置多項式の計算方法を学んだ
- 一般に $t \leq 5$ までの誤り位置多項式の計算方法は知られているが，それより誤り数が大きな誤り位置多項式の計算は複雑になるため，直接計算する方法は知られていない.

本日の講義では，任意の誤り数 t の誤り位置多項式を計算するアルゴリズム，
ユークリッドアルゴリズムを説明する.

ユークリッドアルゴリズム（ユークリッドの互除法）

2個の整数 m, n があるとき，その最大公約数，つまり m と n のどちらも
割り切れるような数の中で最も大きな正整数を見つける.
なお，説明を簡単にするため，以下では $m > n$ とする.

1. m を n で割り，余りを r とする.
2. $r = 0$ なら，アルゴリズム終了． n を答えとして出力する.
3. $m \leftarrow n, n \leftarrow r$ とし，ステップ 1 に戻る.

練習問題 1 （2分）：2つの整数 91 と 65 の最大公約数をユークリッドの互除法で求めよ.

ユークリッドアルゴリズムを詳しく見てみよう

$r_0 = m, r_1 = n$ とすると,

$$
\begin{aligned}
r_0 &= q_1 r_1 + r_2 \\
r_1 &= q_2 r_2 + r_3 \\
r_2 &= q_3 r_3 + r_4 \\
&\vdots \\
r_{k-2} &= q_{k-1} r_{k-1} + r_k \\
r_{k-1} &= q_k r_k + 0
\end{aligned}
$$

となり，r_k が最大公約数となる.

ユークリッドアルゴリズムを詳しく見てみよう

行列表記にすると，ユークリッドアルゴリズムの再帰計算は次のようにも書ける.

$$
\begin{aligned}
\begin{bmatrix} r_0 \\ r_1 \end{bmatrix} &= \begin{bmatrix} q_1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} \\
&= \begin{bmatrix} q_1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} q_2 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} r_2 \\ r_3 \end{bmatrix} \\
&= \quad \vdots \\
&= \begin{bmatrix} q_1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} q_2 & 1 \\ 1 & 0 \end{bmatrix} \cdots \begin{bmatrix} q_k & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} r_k \\ 0 \end{bmatrix}.
\end{aligned}
$$

ユークリッドアルゴリズムで求めたいのは r_k であり，引数は r_0, r_1 であることを思い出すと，r_k を左項に持ってきたい.

ユークリッドアルゴリズムを詳しく見てみよう

$$
\begin{aligned}
\begin{bmatrix} r_k \\ 0 \end{bmatrix} &= \begin{bmatrix} q_k & 1 \\ 1 & 0 \end{bmatrix}^{-1} \cdots \begin{bmatrix} q_2 & 1 \\ 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} q_1 & 1 \\ 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} r_0 \\ r_1 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 1 \\ 1 & -q_k \end{bmatrix} \cdots \begin{bmatrix} 0 & 1 \\ 1 & -q_2 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -q_1 \end{bmatrix} \begin{bmatrix} r_0 \\ r_1 \end{bmatrix} \\
&= \begin{bmatrix} v_0 & w_0 \\ v_1 & w_1 \end{bmatrix} \begin{bmatrix} r_0 \\ r_1 \end{bmatrix}.
\end{aligned}
$$

よって,

$$
r_k = v_0 r_0 + w_0 r_1
$$

となる v_0, w_0 が存在することがわかる.
ユークリッド互除法の計算から，このような v_0, w_0 を求めるアルゴリズムを
拡張ユークリッドアルゴリズムと呼ぶ.

練習問題２(3分)

2つの整数 $r_0 = 91, r_1 = 65$ のときの最大公約数 r_k に対し,

$$r_k = v_0 r_0 + w_0 r_1$$

となる v_0, w_0 を拡張ユークリッドアルゴリズムで求めよ.

拡張ユークリッドアルゴリズムのJuliaでの実装

```
function extended_gcd_iterative(a, b) # a >= b を仮定
    v0, w0, v1, w1 = 1, 0, 0, 1
    while b != 0
        q = a ÷ b
        a, b = b, a % b
        v0, v1 = v1, v0 - q * v1
        w0, w1 = w1, w0 - q * w1
    end
    return (a, v0, w0)
end
```

初期値を

$$\begin{bmatrix} v_0 & w_0 \\ v_1 & w_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

とし，商を計算するたびに更新していけば，最大公約数が導出された際に v_0, w_0 も求まる.

多項式上の拡張ユークリッドアルゴリズム

ここまで，整数の場合を考えてきたが，多項式の場合も同様に考えることができる．
つまり，$r_0(x), r_1(x)$ を多項式として，どちらの多項式も割り切れるような多項式の中で最も次数が大きな多項式 $r_k(x)$ に対して，

$$r_k(x) = v_0(x)r_0(x) + w_0(x)r_1(x)$$

となる $v_0(x), v_1(x)$ が存在し，
それらを求めるアルゴリズムもまた **(多項式上の)拡張ユークリッドアルゴリズム** と呼ぶ．

ユークリッドアルゴリズム

ユークリッドアルゴリズムは誤り位置多項式を導出するアルゴリズムとして，杉山らによって1975年に発明された.

Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, A method for solving key equation for decoding goppa codes, Information and Control, vol.27, issue.1, pp.87-99, 1975.

誤り位置多項式を導出アルゴリズムとしては，ユークリッドアルゴリズム以外にも，Berlekamp-Massey アルゴリズムやPetersonアルゴリズムなどがあるが，ユークリッドアルゴリズムが最もシンプルで理解しやすいことから多くの符号理論の書籍で紹介されている.
また，ユークリッドアルゴリズムの発明に至る当時の環境や様子を記した以下も面白い.
平澤 茂一, 笠原 正雄, ユークリッド復号法, 電子情報通信学会 基礎・境界ソサイエティ Fundamentals Review, 2010, 4 巻, 3 号, p. 183-191

異なる誤り位置多項式(ELP)の定義

誤りの数が s 個のときの誤り位置多項式は以下のように定義される（これまでの定義）

$$\begin{aligned}\sigma'(x) &= (x - X_1)(x - X_2)\cdots(x - X_s) \\ &= \sigma'_0 + \sigma'_1 x + \cdots + \sigma'_{s-1} x^{s-1} + x^s\end{aligned}$$

ただし，$X_j = \alpha^{i_j}$ は j 番目の誤り位置($i_j + 1$ ビット目)に対応する元，σ'_i は誤り位置多項式の係数である.

ここで，**新たな誤り位置多項式**を次のように定義する.

$$\begin{aligned}\sigma(x) &= (1 - X_1 x)(1 - X_2 x)\cdots(1 - X_s x) \\ &= 1 + \sigma_1 x + \cdots + \sigma_{s-1} x^{s-1} + \sigma_s x^s\end{aligned}$$

新たな誤り位置多項式 $\sigma(x)$ の根は，誤り位置に対応する元 X_j の**逆元** X_j^{-1} である.
また多項式としては定数項が 1 となる．$\sigma'(x)$ は最高次 x^s の係数が 1 だった.

シンドローム多項式 $S(x)$

t 誤り訂正可能なBCH符号の符号多項式は $\alpha, \alpha^2, \ldots, \alpha^{2t}$ を根に持つことから，シンドローム多項式 $S(x)$ は次のように定義される．

$$
\begin{aligned}
S(x) &= S_1 + S_2 x + \cdots + S_{2t} x^{2t-1} \\
&= \sum_{l=1}^{2t} S_l x^{l-1}
\end{aligned}
$$

ただし，$S_l = r(\alpha^l) = e(\alpha^l)$ で，$r(x)$ は受信多項式，$e(x)$ は誤り多項式である．

シンドローム多項式 $S(x)$

シンドローム多項式 $S(x)$ は次のように書くこともできる.

$$
\begin{aligned}
S(x) &= \sum_{l=1}^{2t} S_l x^{l-1} \\
&= \sum_{l=1}^{2t} \sum_{j=1}^{s} X_j^l x^{l-1} \\
&= \sum_{j=1}^{s} \sum_{l=1}^{2t} X_j (X_j x)^{l-1} \\
&= \sum_{j=1}^{s} \frac{X_j}{1 - X_j x} \mod x^{2t}
\end{aligned}
$$

ただし，$X_j = \alpha^{i_j}$ は j 番目の誤り位置($i_j + 1$ ビット目)に対応する元で s は誤りの個数.

最後の等式

等比級数の恒等式を形式冪級数の世界で用い，次数が $2t$ 以上の項は $\mod x^{2t}$ でゼロと同一視することにより，

$$\sum_{l=1}^{2t} X_j \left(X_j x\right)^{l-1} \equiv \frac{X_j}{1 - X_j x} \pmod{x^{2t}}$$

が成り立つ．

シンドローム多項式$S(x)$は次数$2t-1$までしか係数を使わないので，$\left(X_j x\right)^{2t}$以上の高次項は計算結果に寄与しないためこの変形が可能．

最後の等式変形の詳細

左項の和は等比級数なので，まず有限和の閉形式を書き下す.

$$\sum_{l=1}^{2t} X_j \left(X_j x\right)^{l-1} = X_j \sum_{l=0}^{2t-1} \left(X_j x\right)^{l} = X_j \frac{1-\left(X_j x\right)^{2t}}{1-X_j x}.$$

ここで右辺を二つに分けると,

$$X_j \frac{1}{1-X_j x} - X_j \frac{\left(X_j x\right)^{2t}}{1-X_j x} = \frac{X_j}{1-X_j x} - \frac{X_j^{2t+1} x^{2t}}{1-X_j x}.$$

ここで，$\mod x^{2t}$ をすることで，次数が $2t$ 以上の項は取り除かれ

$$\sum_{l=1}^{2t} X_j \left(X_j x\right)^{l-1} \equiv \frac{X_j}{1-X_j x} \pmod{x^{2t}}$$

となり，右辺が得られる.

小さな具体例（$t = 2$）：

$$\sum_{l=1}^{4} X\,(Xx)^{\,l-1} = X + X^2x + X^3x^2 + X^4x^3\,,$$

一方,

$$\frac{X}{1-Xx} = X + X^2x + X^3x^2 + X^4x^3 + X^5x^4 + \cdots\,.$$

$\bmod x^4$ではx^4以上を捨てるので,

$$\frac{X}{1-Xx} \equiv X + X^2x + X^3x^2 + X^4x^3 \pmod{x^4}$$

となり，一致する.

ELP計算の続き

誤り位置多項式 $\sigma(x)$ とシンドローム多項式 $S(x)$ との積を考える.

$$\sigma(x)S(x) = (1 - X_1 x)(1 - X_2 x)\cdots(1 - X_s x)\sum_{j=1}^{s}\frac{X_j}{1 - X_j x} \mod x^{2t}$$

$$= \sum_{j=1}^{s} X_j \prod_{\substack{k=1 \\ k\neq j}}^{s}(1 - X_k x) \mod x^{2t}$$

右項を $\Omega(x)$ とすると，左項を x^{2t} で割った剰余とみなせるので,

$$\sigma(x)S(x) = q(x)x^{2t} + \Omega(x)$$

とかける．$\Omega(x)$ が左項になるよう式を整理すると,

$$\Omega(x) = q(x)x^{2t} + \sigma(x)S(x)$$

となることから，$x^{2t}, S(x)$ から拡張ユークリッドアルゴリズムを用いて $\sigma(x)$ が求まる.

ユークリッドアルゴリズム

入力 $S(x), t$

1. （初期化）$r_0(x) \leftarrow x^{2t}, r_1(x) \leftarrow S(x), w_0(x) = 0, w_1(x) = 1, i \leftarrow 1$ とする.
2. $r_{i-1}(x) = q_i(x) r_i(x) + r_{i+1}(x)$ を計算する.
3. $w_{i+1}(x) = -q_i(x) w_i(x) + w_{i-1}$
4. $\deg r_{i+1}(x) \geq t$ ならば $i \leftarrow i + 1$ としてステップ 2 に戻る.
5. $w_{i+1}(x)$ の定数項 $w_{i+1,0}$ を 1 にするように $\sigma(x) = w_{i+1,0}^{-1}\, w_{i+1}(x)$ を出力する.

練習問題３（１５分）

$GF(2^4)$ の原始元 α の最小多項式 を $\mu_\alpha(x) = 1 + x + x^4$ とする．生成多項式
$g(x) = \mu_\alpha \mu_{\alpha^3} \mu_{\alpha^5}$ で定義された長さ 15 の3誤り訂正2元BCH符号によって符号化された
符号多項式が，通信路を通って受信多項式 $r(x) = x + x^3 + x^5$ となったとき，ユークリッドアルゴリズムを用いて誤り位置多項式を求めよ．

練習問題3デモ概要

`jl/10-ex3-demo.jl` では $GF(2^4)$ を $\mu_\alpha(x) = 1 + x + x^4$ で構成し，$t = 3$ のシンドローム多項式 $S(x)$ と x^{2t} を初期条件として拡張ユークリッドアルゴリズムを逐次表示しながら実行した.
受信多項式 $r(x) = x + x^3 + x^5$ から計算した $S(x)$ の係数 $\{S_i\}$ をまず出力し，続いて各ステップの被除数 $r_{i-1}(x)$，除数 $r_i(x)$，商 $q_i(x)$，剰余 $r_{i+1}(x)$，更新後の $w_i(x)$ をその次数付きで一覧表示して手計算を追えるようにしている.

Julia実装

```
function euclid_with_trace(Sx, t)
    base_type = typeof(α)
    head_coeffs = zeros(base_type, 2t + 1)
    head_coeffs[end] = base_type(1)
    r_prev = Polynomial(head_coeffs, "x")
    r_curr = Sx
    w_prev = Polynomial([base_type(0)], "x")
    w_curr = Polynomial([base_type(1)], "x")
    println("=== 初期化 ===")
    show_poly("r₀(x) = x^{2t}", r_prev)
    show_poly("r₁(x) = S(x)", r_curr)
    show_poly("w₀(x)", w_prev)
    show_poly("w₁(x)", w_curr)
    step = 1
    while degree(r_curr) ≥ t && !iszero(r_curr)
        q, r = divrem(r_prev, r_curr)
        println("\n=== Step $step: r_{step-1}(x) ÷ r_{step}(x) ===")
        show_poly("r_{step-1}(x)", r_prev)
        show_poly("r_{step}(x)", r_curr)
        show_poly("q_$step(x)", q)
        show_poly("r_{step+1}(x)", r)
        r_prev, r_curr = r_curr, r
        w_prev, w_curr = w_curr, w_prev - q * w_curr
        show_poly("w_{step}(x)", w_prev)
        show_poly("w_{step+1}(x)", w_curr)
        step += 1
    end
    σ = w_curr / w_curr.coeffs[1]
    println("\n=== 正規化 ===")
    show_poly("σ(x) before norm", w_curr)
    show_poly("σ(x)", σ)
    Ω = r_curr
    show_poly("Ω(x)", Ω)
    return σ, Ω
end
```

練習問題3デモ出力ログ

```
=== 練習問題3 デモ ===
GF(2^4) 生成多項式: x^4 + x + 1, t = 3

受信多項式 r(x): 1*x + 1*x^3 + 1*x^5

=== シンドローム計算 ===
S(x) (deg 5): [α^6, α^12, α^4, α^9, 0, α^8]
=== 初期化 ===
r₀(x) = x^{2t} (deg 6): [0, 0, 0, 0, 0, 0, α^0]
r₁(x) = S(x) (deg 5): [α^6, α^12, α^4, α^9, 0, α^8]
w₀(x) (deg -1): []
w₁(x) (deg 0): [α^0]

=== Step 1: r_{step-1}(x) ÷ r_{step}(x) ===
r_{step-1}(x) (deg 6): [0, 0, 0, 0, 0, 0, α^0]
r_{step}(x) (deg 5): [α^6, α^12, α^4, α^9, 0, α^8]
q_1(x) (deg 1): [0, α^7]
r_{step+1}(x) (deg 4): [0, α^13, α^4, α^11, α^1]
w_{step}(x) (deg 0): [α^0]
w_{step+1}(x) (deg 1): [0, α^7]

=== Step 2: r_{step-1}(x) ÷ r_{step}(x) ===
r_{step-1}(x) (deg 5): [α^6, α^12, α^4, α^9, 0, α^8]
r_{step}(x) (deg 4): [0, α^13, α^4, α^11, α^1]
q_2(x) (deg 1): [α^2, α^7]
r_{step+1}(x) (deg 3): [α^6, α^11, α^14, α^14]
w_{step}(x) (deg 1): [0, α^7]
w_{step+1}(x) (deg 2): [α^0, α^9, α^14]
```

```
=== Step 3: r_{step-1}(x) ÷ r_{step}(x) ===
r_{step-1}(x) (deg 4): [0, α^13, α^4, α^11, α^1]
r_{step}(x) (deg 3): [α^6, α^11, α^14, α^14]
q_3(x) (deg 1): [α^7, α^2]
r_{step+1}(x) (deg 2): [α^13, 0, α^1]
w_{step}(x) (deg 2): [α^0, α^9, α^14]
w_{step+1}(x) (deg 3): [α^7, α^13, α^1, α^1]

=== 正規化 ===
σ(x) before norm (deg 3): [α^7, α^13, α^1, α^1]
σ(x) (deg 3): [α^0, α^6, α^9, α^9]
Ω(x) (deg 2): [α^13, 0, α^1]

=== 誤り位置探索 ===
σ(α^10) = 0 → X = α^5 → ビット位置 6
σ(α^12) = 0 → X = α^3 → ビット位置 4
σ(α^14) = 0 → X = α^1 → ビット位置 2

推定誤り位置指数 (0-index): [5, 3, 1]
```

解法の流れ

ログのように $\deg r_i(x) \geq t$ を満たす間はユークリッド除算を繰り返し，商で $w_i(x)$ を線形更新しながら $\sigma(x)$ を構成する.
最終ステップで $w_{i+1}(x)$ の定数項を 1 に正規化すると $\sigma(x) = 1 + \alpha^6 x + \alpha^9 x^2 + \alpha^9 x^3$ が得られ，$\sigma(\alpha^{10}) = \sigma(\alpha^{12}) = \sigma(\alpha^{14}) = 0$ となるため誤り位置はビット番号 $\{2, 4, 6\}$（0 始まりでは $\{1, 3, 5\}$）と読み取れる.